

Programmation Procédurale

INF147


Automne 2023

Présenté par :
El Hachemi Alikacem

Séance 1 : Introduction



1



2

Plan

- Programmation : notions de bases et définitions
- Types fondamentaux en C
- Variables
- Opérateurs
- Instruction d'affectation (←)
- Entrées/sorties simples
- Instruction de sélection (l'alternative)
- Instruction itérative –la boucle while

*Présentation basée principalement sur les notes de cours élaborées par M. Pierre Belisle
Maître d'enseignement au SEG*

El Hachemi Alikacem

INF147

2

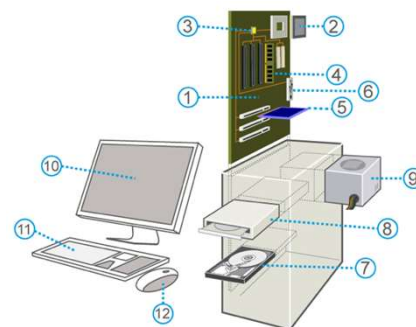
L'ordinateur

Un **ordinateur** est une machine programmable de traitement de l'information, commandée par des programmes stockés en mémoire, qui accepte des données structurées, les traite selon des règles définies et produit automatiquement un résultat en sortie (cf. OQLF).

Un ordinateur est constitué d'une **unité centrale** (pour exécuter les programmes), d'une **mémoire centrale** (pour stocker les données et les logiciels) et de **périphériques**, comme le clavier, la souris, l'écran et les haut-parleurs (pour communiquer avec l'utilisateur).

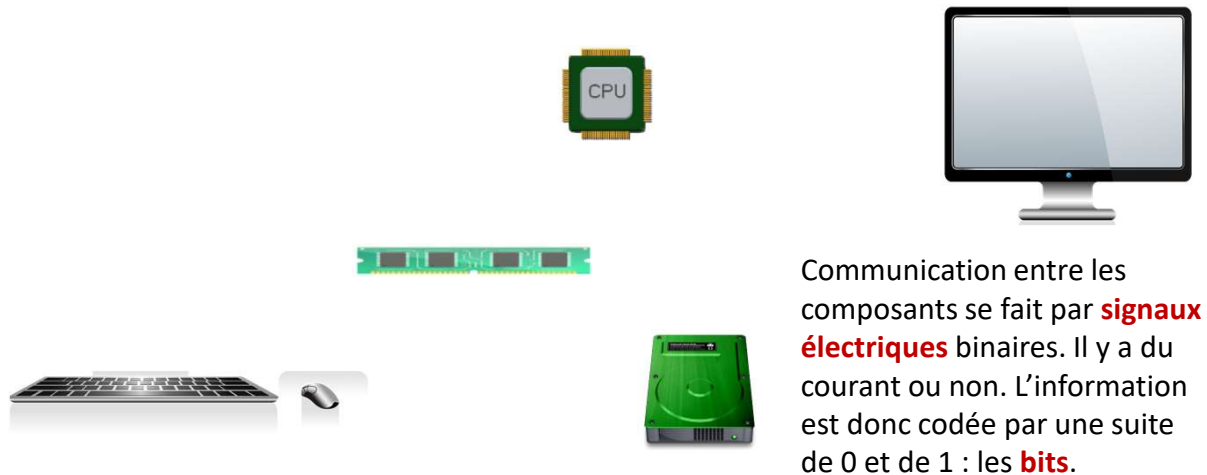
Composantes d'un ordinateur

1	Carte mère	Liaison des éléments
2	Processeur	Unité de traitements
3	Bus	Communication inter-composants
4	Mémoire (RAM)	Stockage temporaire
5	Carte graphique	Produire l'affichage
6	Entrées/Sorties	Communication externe
7	Disque Dur	Stockage permanent
8	Lecteur de disque	lecture/gravure de disque
9	Alimentation	Fournit l'énergie
10	Moniteur	Visualisation
11	Clavier	Saisie de texte
12	Souris	Dispositif de pointage



5

Fonctionnement d'un ordinateur (schématisation)



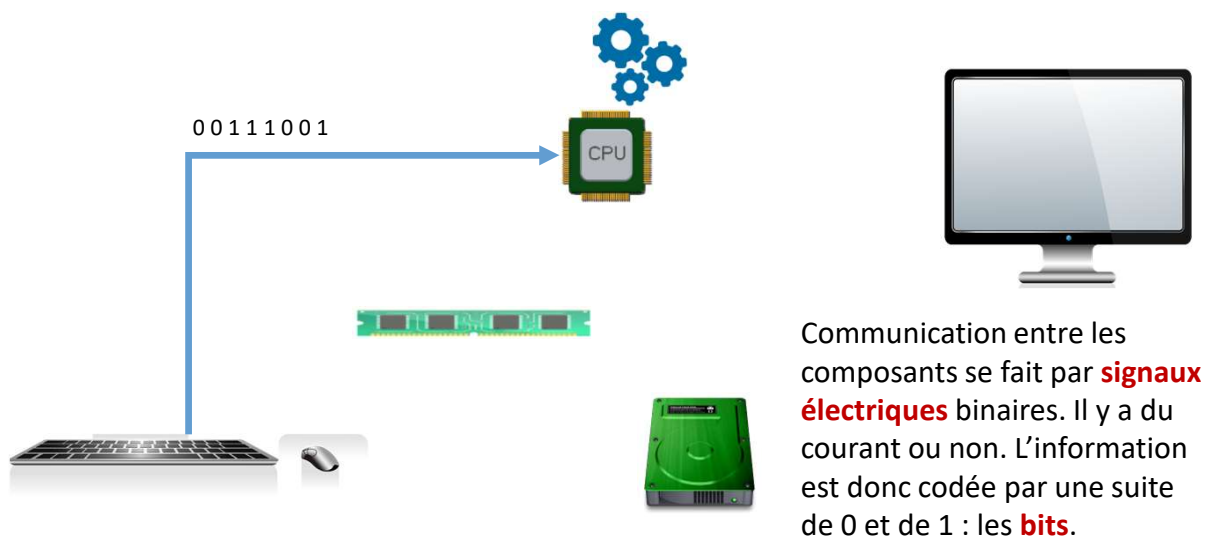
El Hachemi Alikacem

INF147

5

6

Fonctionnement d'un ordinateur



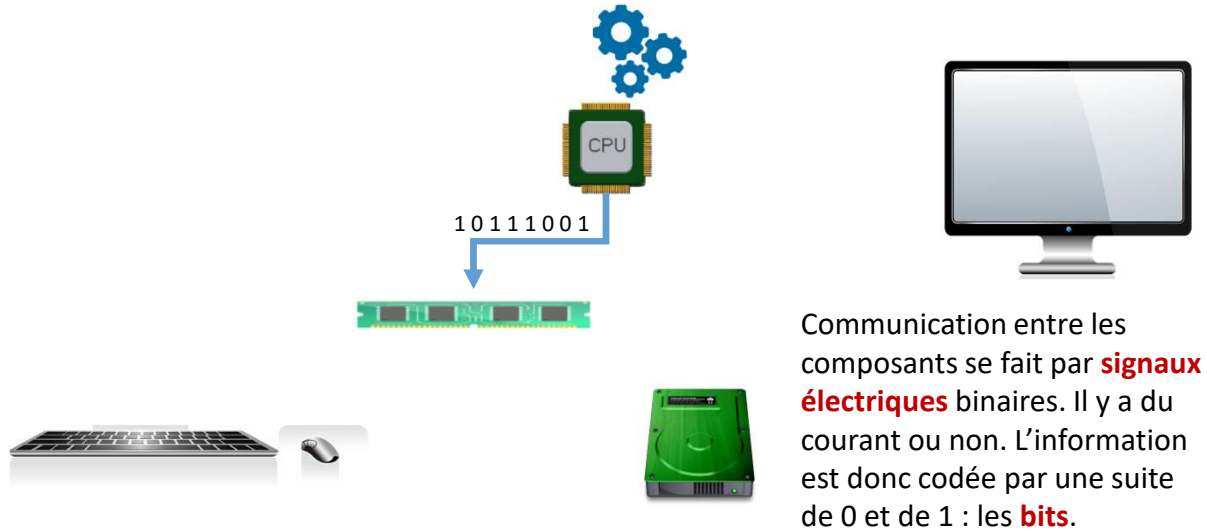
El Hachemi Alikacem

INF147

6

7

Fonctionnement d'un ordinateur



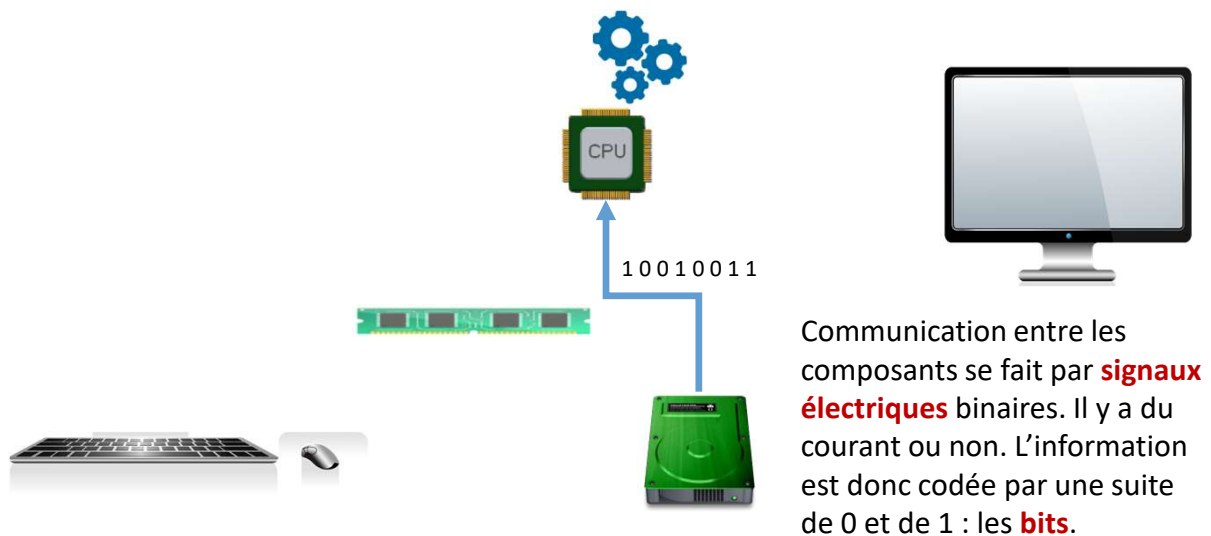
El Hachemi Alikacem

INF147

7

8

Fonctionnement d'un ordinateur

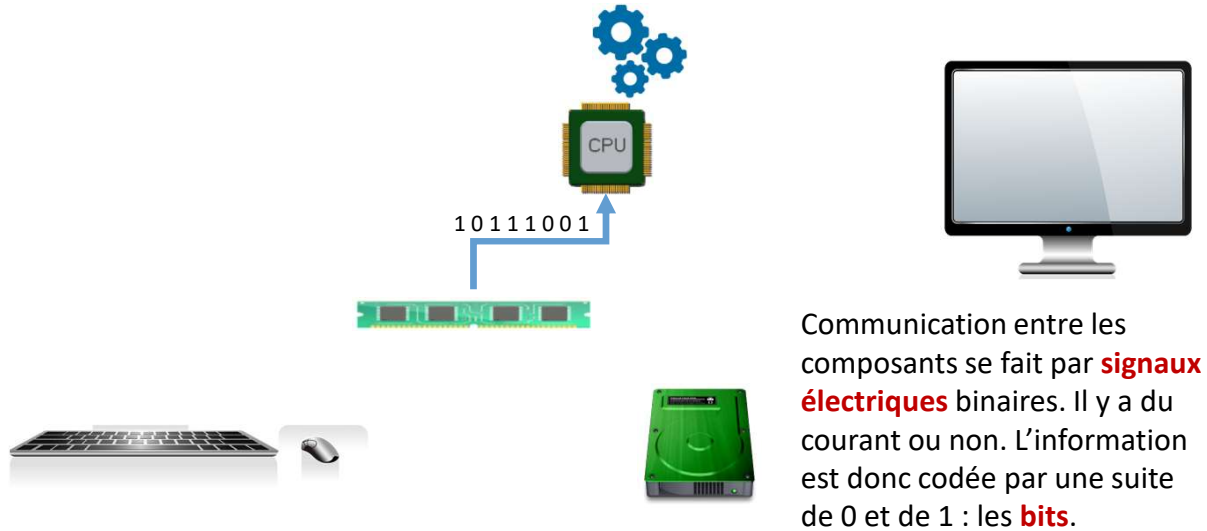


El Hachemi Alikacem

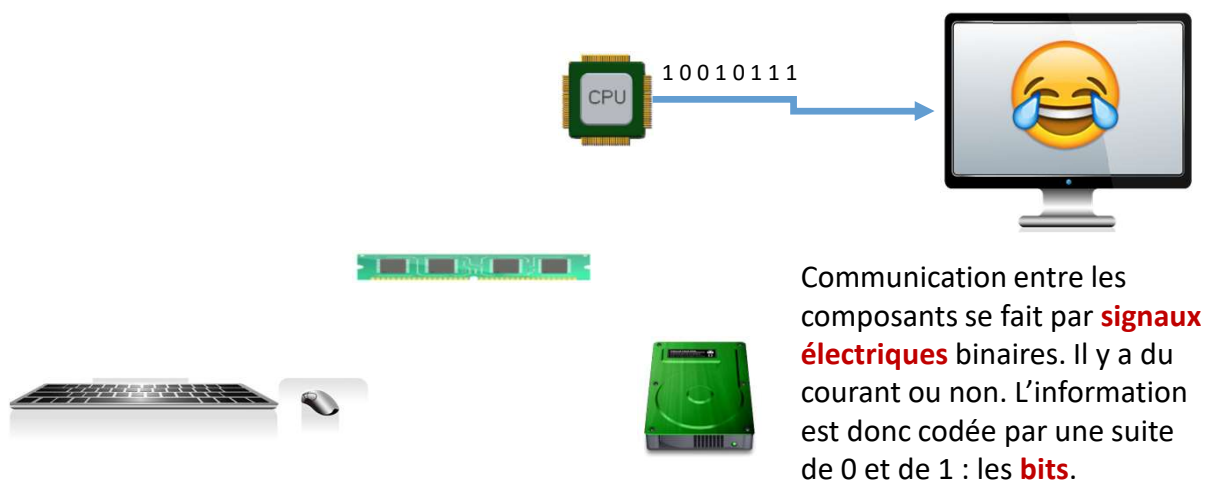
INF147

8

Fonctionnement d'un ordinateur



Fonctionnement d'un ordinateur



À quoi ça sert ?

- Juste à faire des calculs ... mais très rapidement. Les processeurs «grand public» actuels font 4 000 000 000 **d'instructions machines** par seconde !!!
- Une **instruction machine** est une opération élémentaire qu'un **programme** demande à un processeur d'effectuer. C'est l'ordre le plus basique que peut comprendre un ordinateur.
 - Instructions de transfert : Déplacer de l'information
 - Instructions arithmétiques : Addition, soustraction, division, multiplication
 - Instructions logiques : OU, ET, NON, XOR
 - Instructions de branchement : Passer d'un programme à un autre
- Un **programme** est une succession d'instructions machines

11

C'est quoi un programme informatique ?

- Définition Wikipédia :

Un **programme informatique** indique à un ordinateur ce qu'il devrait faire. Il s'agit d'un ensemble d'instructions qui doivent être exécutées dans un certain ordre par un processeur.

12

ETS

13

C'est quoi un interprète

- Définition Wikipédia :
 - En informatique, un **interprète** (parfois appelé, à tort, *interpréteur* par mauvaise traduction de l'anglais) est un outil ayant pour tâche d'analyser, de traduire, et d'exécuter un programme écrit dans un langage informatique
- Le cycle d'un interprète est le suivant:
 - lire et analyser une instruction (ou expression) ;
 - si l'instruction est syntaxiquement correcte, l'exécuter (ou évaluer l'expression);
 - passer à l'instruction suivante;
- Avantage
 - Plus vite à développer pour le programmeur
- Inconvénient
 - Plus lent à l'exécution

L'interprétation (analyse, traduction en code exécutable puis exécution) d'un programme se fait instruction par instruction.

El Hachemi Alikacem

INF147

13

ETS

14

C'est quoi un compilateur ?

- Définition Wikipédia :

Un **compilateur** est un programme informatique qui traduit un langage, le *langage source*, en un autre, appelé le *langage cible*...En pratique, un compilateur sert le plus souvent à traduire un *code source* écrit dans un langage de programmation en un autre langage, habituellement un langage d'assemblage ou un langage machine. Le programme en langage machine produit par un compilateur est appelé code objet.
- Compilateur
 - La traduction de tout le code est faite avant la première exécution
- Avantage
 - Plus vite à l'exécution
- Inconvénient
 - Plus lent et complexe à développer pour le programmeur

Dans la compilation, le programme est traduit en code exécutable dans sa totalité avant son exécution

El Hachemi Alikacem

INF147

14

ETS

15

C'est quoi un langage de programmation ?

- Définition Wikipédia :
Un langage de programmation permet d'écrire des programmes. L'activité de rédaction du code source d'un logiciel est nommée programmation.
- Il existe des dizaines de langages. À titre d'exemples :
 - ▢ C, C++, Java, Visual Basic, Ada, Delphi, Lisp, Prolog, Sml, Html, Xml, Sql, ...

El Hachemi Alikacem

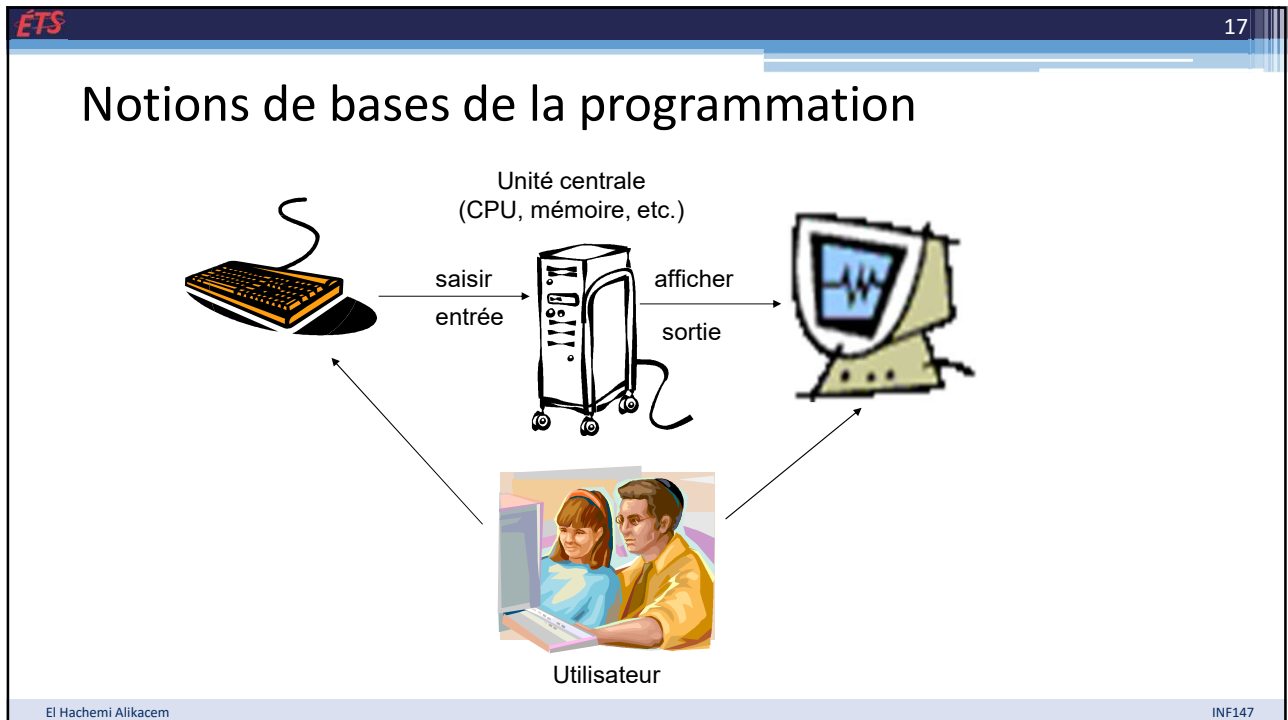
INF147

15

16

Principe de programmation de base

16



17

18

La programmation

La **programmation** correspond à l'ensemble des activités techniques reliées à l'élaboration d'un programme informatique. Elle comprend donc les activités de **conception**, **d'écriture**, de **test** et de **maintenance** de programmes (cf. OQLF).

Un programmeur doit être en mesure de :

- ▢ trouver une solution à un problème;
- ▢ exprimer cette solution de façon structurée;
- ▢ traduire cette solution en un langage compréhensible par la machine;
- ▢ tester le programme.

El Hachemi Alikacem INF147

18

ETS

19

Principe de programmation de base

- Instruction
 - Commande reconnue par un langage de programmation et exécutable par l'ordinateur.
- Programme principal
 - Bloc d'instructions englobant les instructions à exécuter. Ce bloc a un début et une fin.
- Entrées/sorties simples
 - Entrée clavier : lire ou saisir
 - Sortie écran : écrire ou afficher

El Hachemi Aliakem

INF147

19

20

Écrire un programme

Écrire un programme : c'est écrire du texte dans un **environnement de développement** en utilisant un **langage de programmation**.

Un langage de programmation est composé dispose de :

- Ensemble de mots clés (**If**, **For**, **End For** ...)
- Liste de Symboles (+, -, & , '...)

Un langage est défini par une syntaxe : (**If** $x < 1$ **Then**). Elle nous indique comment constituer une instruction :

- Syntaxe

El Hachemi Aliakem

INF147

20

ÉTS

21

Principe de programmation de base

- Dans un programme, on manipule des données qui ont un type. Par exemples :
 - Entier, réel, caractère et booléen
 - *C'est les types fondamentaux*
- Variable
 - Espace mémoire modifiable qui doit être réservé par le programmeur
 - Doit avoir un type spécifié
 - Doit avoir un nom.
- Instruction d'affectation
 - Permet de mettre une valeur directement dans une variable
 - Ex: *salaire* ← 2,45 (En C c'est l'opérateur =)

El Hachemi Aliakem

INF147

21

ÉTS

22

Principe de programmation de base

- Plusieurs opérateurs sont utilisés en programmation. Par exemples :
 - Arithmétiques :
 - Addition, soustraction, division, multiplication, modulo (reste de la division entière)
 - Booléens :
 - et, ou, non, ou exclusif
 - Relationnels :
 - Plus petit, plus grand, égal, différent, plus petit ou égal, plus grand ou égal.

El Hachemi Aliakem

INF147

22

ÉTS

23

Programmation : exemple

Essayons d'élaborer (informellement) un programme pour le calcul du salaire brut. Sachant que le salaire brut est le nombre d'heures travaillées fois le taux horaire.

Les instructions constituant le programme de calcul de salaire brut :

Début

- Déclaration de variables pour conserver un taux horaire, un nombre d'heures travaillées et un salaire,
 - Toutes ces variables seront de type réel
- Afficher un message de sollicitation pour entrer le taux horaire
- Instruction pour saisir le taux horaire au clavier
- Afficher un message de sollicitation pour le nombre d'heures travaillées
- Instruction pour saisir le nombre d'heures travaillées au clavier
- Calcul du salaire et brut et l'affecter à la variable : Salaire \leftarrow nombre d'heures travaillées * taux horaire
- Afficher le salaire

Fin

El Hachemi Alikacem

INF147

23

ÉTS

24

Algorithme - définition

- C'est la stratégie qu'on prend pour résoudre un problème informatique (il y a d'autres définitions).
- Un programme est une traduction, dans un langage de programmation, d'un algorithme.
- Il peut y avoir plusieurs algorithmes qui mènent à un programme exécutable identique.

El Hachemi Alikacem

INF147

24

Programmation en C

25

ÉTS

26

Programmation en C

- En C, il existe plusieurs types fondamentaux, tels que :
 - ▢ Entiers : **short int** (8 bits), **int** (16 ou 32 bits), **long** (32 bits)
 - ▢ Réels : **float** (32 bits), **double** (64 bits), **long double** (64 ou 80 bits)
 - ▢ Caractères : **char** (8 bits), **int** (16 bits)
 - ▢ Les booléens sont représentés par 0 (faux) et 1 (vrai)
- Tous ces types peuvent être préfixés de *unsigned* (non signés)

El Hachemi Aliakem

INF147

26

Type	Longueur	Valeurs Min/Max
unsigned char	8 bits	0 à 255
char	8 bits	-128 à 127
unsigned short int	16 bits	0 à 65 535
short int	16 bits	-32 768 à 32 767
unsigned int	32 bits	0 à 4 294 967 295
int	32 bits	-2 147 483 648 à 2 147 483 647
unsigned long	32 bits	0 à 4 294 967 295
long	32 bits	-2 147 483 648 à 2 147 483 647
float	32 bits	$3.4 * 10^{-38}$ à $3.4 * 10^{38}$
double	64 bits	$1.8 * 10^{-308}$ à $1.8 * 10^{+308}$

El Hachemi Aliakem

INF147

27

Variable
<ul style="list-style-type: none"> • Une variable est définie par un identificateur (ou nom de variable) qui doit respecter des règles : <ul style="list-style-type: none"> □ Un identificateur est composé de lettres, de chiffres et de symboles □ Un identificateur doit obligatoirement commencer par une lettre □ Le seul symbole permis est le petit souligné (_) □ Le langage est sensible à la case (minuscule/majuscule) ➤ Exemples d'identificateurs valides : Nom, prenom, nom_joueur1, nom_joueur2 ➤ Exemples d'identificateurs invalides : 1nom, #toto, ceci n'est pas valide ➤ Recommandation : n'utiliser pas les lettres avec accent

El Hachemi Aliakem

INF147

28

ETS

29

Déclaration d'une variable

En C, toute variable doit être déclarée avant son utilisation. Pour cela, on indique le type de la variable suivi du nom de la variable (ou identificateur) suivi, optionnellement, d'une valeur initiale, sans oublier le **point-virgule à la fin**.

- La forme générale pour déclarer une variable est : `Type nom_variable [= valeur] ;`
- Exemples :
 - ▢ `int age;`
 - ▢ `double salaire;`
 - ▢ `int somme = 0;`
- **Le choix du type de la variable doit prendre en compte la nature des valeurs qui lui seront affectées**

El Hachemi Aliakem

INF147

29

ETS

30

Les opérateurs

- C offre de nombreux opérateurs. Parmi eux :
 - ▢ Arithmétiques : `+`, `-`, `*`, `/`, `%` (modulo d'une division entière)
 - ▢ Booléens : `&&` (et), `||` (ou), `!` (non), `^` (ou exclusif sur les bits)
 - ▢ Comparaisons : `<`, `>`, `<=`, `>=`, `==`, `!=`

El Hachemi Aliakem

INF147

30

ETS
31

Expression

- Une expression est une construction syntaxique qui a un **type** et une **valeur** (on parle d'évaluation de l'expression).
- Une expression peut être :
 - ▢ Une constante (une valeur numérique, un caractère, etc.), par exemple 3.14
 - ▢ Une variable, par exemple **compteur**
 - ▢ Une opération (arithmétique, booléenne, etc.), par exemples : $(a+b)/2$, $a==12$
 - ▢ Un appel de fonction, par exemple **calculMoyenne()**
 - ▢ Etc.
- Noter que l'instruction d'affectation est aussi une expression dont la valeur est la valeur affectée à la variable, par exemple : **total = 123 ;** la valeur de l'affectation est 123

El Hachemi Aliakem
INF147

31

ETS
32

Affectation

Instruction d'affectation consiste à attribuer le résultat d'une expression à une variable. La forme générale est la suivante :

$$\text{variable} = \text{expression} ;$$

Le terme de gauche

↑

Opérateur d'affectation

↑

Le terme de droite

↑

- L'exécution de l'affectation se fait en deux étapes :
 1. Évaluation de l'expression (le terme de droite)
 2. Affectation du résultat de l'évaluation à la variable (le terme de gauche)
- Les types de l'expression et de la variable doivent être identiques, sinon « compatibles »
- Si les types ne sont pas identiques, une conversion de type implicite est effectuée : la valeur de l'expression est convertie dans le type de la variable

El Hachemi Aliakem
INF147

32

ETS
33

Affectation

- Exemples :
 - ▢ `distance = 14 ;`
 - ▢ `age = valeur ;`
 - ▢ `total = (a+b) / 2.25 ;`
 - ▢ `x = x + 1; ←` Cette affectation permet d'incrémenter de 1 la valeur de la variable x
- Affectations abrégées :

Affectation classique	abréviation
<code>x = x - 3</code>	<code>x -= 3</code>
<code>x = x * 2</code>	<code>x *= 2</code>
<code>x = x % 2</code>	<code>x %= 2</code>
<code>x = x / 2</code>	<code>x /= 2</code>
<code>x = x + 1</code>	<code>x++</code>
<code>x = x - 1</code>	<code>x--</code>

El Hachemi Alikacem
INF147

33

ETS
34

Affectation

- Différence entre `x++` et `++x`
 - ▢ `x++` : post incrémentation - L'expression entière est évaluée avant l'incrément
 - ▢ `++x` : pré incrémentation - L'incrément se fait avant l'évaluation de l'expression

Exemple 1 :

```
int tour = 1;
printf("%d", tour++);
```

Affiche 1 puis réalise l'instruction : `tour = tour + 1`

Exemple 2 :

```
int tour = 1;
printf("%d", ++tour);
```

Réalise l'instruction : `tour = tour + 1`, puis affiche 2

El Hachemi Alikacem
INF147

34

Affectation

- En C, l'instruction d'affectation est aussi une expression dont la valeur est la valeur affectée à la variable.

Par exemple : `total = 123` ; la valeur de l'affectation est 123

Affichage sur la console : la fonction `printf`

- La fonction `printf` permet d'afficher des informations (résultats numériques, message, etc.) sur la console, en utilisant des **symboles** indiquant le format d'affichage.
- Note : le format d'affichage dépend du type de l'information à afficher

Exemples de format :

Code format	Type de l'argument à afficher	Format d'affichage et exemples
<code>%c</code>	caractère	Ex : a G ? +
<code>%hd</code>	entier <i>short int</i> (ou caractère) signé	Base 10. Ex : -12
<code>%hu</code>	entier <i>short int</i> (ou caractère) non signé	Base 10. Ex : 463
<code>%hX</code>	pour afficher en hexadécimal (<i>short</i> ou <i>char</i>)	Hexadécimal. Ex : 9A0F
<code>%ld</code>	entier <i>long int</i> signé	Base 10. Ex : -1289
<code>%lu</code>	entier <i>long int</i> non signé	Base 10. Ex : 46399
<code>%lX</code>	entier <i>long int</i> en hexadécimal	Hexadécimal. Ex : B4E98A0F
<code>%d</code>	entier <i>int</i>	Base 10. Ex : -546
<code>%X</code>	pour afficher un <i>int</i> en hexadécimal	Hexadécimal. Ex : 9A0F
<code>%lf</code>	Réel double précision (<i>double</i>)	Virgule flottante. Ex : -3.141592 Avec exposant. Ex : -1.450000e-7
<code>%f</code>	Réel simple précision (<i>float</i>)	Virgule flottante. Ex : -3.141592 Avec exposant. Ex : -1.450000e-7
<code>%s</code>	Chaîne de caractères	Ex : bonjour !

Les codes formats

Source : https://public.iutenligne.net/informatique/algorithmie-et-programmation/priou/LangageC/61_affichage_laide_de_la_fonction_printf.html

ÉTS

37

Affichage sur la console : la fonction `printf`

Exemples :

- Afficher un message :

```
printf("bonjour tout le monde");
```
- Afficher un message et la valeur d'une variable :

```
printf("Votre salaire est : %lf", salaire);
```

 - Le format d'affichage `%lf` correspond à la variable `salaire`
- Afficher plusieurs variables :

```
printf("Votre salaire est : %lf \n Pour le taux %lf", salaire, taux);
```
- Un autre exemple :

```
printf("%c", 65); // affiche 'A' - 65 correspond au code ASCII de 'A'
```

El Hachemi Aliakem

INF147

37

ÉTS

38

La saisie au clavier : la fonction `scanf`

Pour la saisie au clavier, on doit spécifier le type des valeurs à lire à l'aide d'un format de lecture

- Format de lecture :
 - `%d` : lecture d'un entier
 - `%lf` : lecture d'un réel
 - `%c` : lecture d'un caractère
- La fonction `scanf` : `scanf("format de lecture" , &nom_variable);`

Exemple :

```
printf("Entrez votre taux horaire svp : "); // affiche un message d'indication
scanf ("%lf", &taux_horaire);             // lecture au clavier d'une valeur réelle
```

- Le symbole `'&'` est vital devant le nom de la variable (pour l'instant)

El Hachemi Aliakem

INF147

38

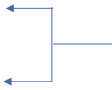
ETS
39


Structure du programme simple


- Structure d'un programme simple en C

```

#include ...
#define ...
//entête du programme principal
int main(void)
{
    Déclaration de variables;
    Instructions;
    return EXIT_SUCCESS;
}
  
```


 Section des commandes (ou directives) au préprocesseur


 //début d'un bloc de code


 //fin du bloc

El Hachemi Alikacem
INF147

39

ETS
40

Commande au préprocesseur

Les commandes au préprocesseur sont des commandes qui sont exécutées avant la compilation :

- La directive `#include` : permet d'inclure des bibliothèques de fonctions utilitaires nécessaires.

Exemples :

```

□ #include <stdio.h>    // permet d'utiliser printf et scanf
□ #include <stdlib.h>   // permet d'utiliser EXIT_SUCCESS
□ #include <math.h>     // permet d'utiliser cos, sin, sqrt, ...
□ Etc.                 // nous en verrons d'autres plus tard
  
```

El Hachemi Alikacem
INF147

40

Commande au préprocesseur

- La directive `#define` : permet de donner des noms à des valeurs constantes du programme

Exemples :

```
#define TRUE 1
#define FALSE 0
#define MAX_HEURE 40.0
```

- Notez que :
 - Les noms des constantes seront toujours en majuscules
 - Dans une commande, il n'y a pas de ';' à la fin

41

Programme de calcul de salaire en C

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    double taux;
    double nb_heures;
    double salaire;

    printf("\n\nEntrez le taux horaire svp : ");
    scanf("%lf", &taux);

    printf("\n\nEntrez le nombre d'heures travaillées : ");
    scanf("%lf", &nb_heures);

    salaire = taux * nb_heures;
    printf("\n\nVotre salaire de brut est : %lf\n", salaire);

    return EXIT_SUCCESS;
}
```

Diagramme d'annotation du code :

- Commandes au préprocesseur** : pointe vers `#include <stdio.h>` et `#include <stdlib.h>`.
- Entête et début de bloc** : pointe vers `int main(void) {`.
- Déclarations de variables** : pointe vers `double taux;`, `double nb_heures;` et `double salaire;`.
- Message de sollicitation à l'écran** : pointe vers `printf("\n\nEntrez le taux horaire svp : ");`.
- Saisie au clavier et assignation à la variable** : pointe vers `scanf("%lf", &taux);` et `scanf("%lf", &nb_heures);`.
- Calcul et affichage du salaire** : pointe vers `salaire = taux * nb_heures;` et `printf("\n\nVotre salaire de brut est : %lf\n", salaire);`.
- Retour d'un résultat et fin de bloc** : pointe vers `return EXIT_SUCCESS;` et `}`.

42

Les commentaires

- Le commentaire est un texte inséré par le développeur afin d'aider à la compréhension du code

- Il y a deux façons d'écrire un commentaire :

1. Commentaire multilignes débute par `/*` et se termine par `*/`

Exemple :

```
/* Ceci est un commentaire,
   la suite est sur la deuxième ligne */
```

- Note : on ne peut pas imbriquer des commentaires

2. Commentaire sur une ligne débute par `//` et jusqu'à la fin de la ligne

Exemple :

```
// Commentaire sur une seule ligne
```

Code avec les commentaires

Commentaire multilignes
(en bloc avec `/*` et `*/`)

Commentaires sur une
seule ligne (avec `//`)

```
#include <stdio.h>
#include <stdlib.h>
/*
 * Programme qui calcule un salaire brut à partir
 * d'un taux horaire et d'un nombre d'heures travaillées
 */
int main(void) {
    double taux;           // sert à stocker le taux horaire
    double nb_heures;      // sert à stocker le nombre d'heures
    double salaire;        // sert au calcul du salaire

    //solicitation du taux horaire
    printf("\n\nEntrez le taux horaire svp : ");
    scanf("%lf", &taux);

    //solicitation du nombre d'heures travaillées
    printf("\n\nEntrez le nombre d'heures travaillées : ");
    scanf("%lf", &nb_heures);

    //calcul et affichage du salaire
    salaire = taux * nb_heures;
    printf("\n\nVotre salaire brut est : %lf", salaire);

    return EXIT_SUCCESS;
}
```

La sélection

45

Sélection

- L'instruction de sélection (ou structure d'alternative) permet qu'une instruction (ou un bloc d'instructions) soit exécutée selon une condition
- La condition est une expression « habituellement » booléenne, évaluée à **Vrai** ou **Faux**
 - Exemple en pseudo-code :


```

Si j'ai assez d'argent  ← la condition
    je vais au cinéma  ← l'instruction
Fin
          
```
- La condition est évaluée en premier, si le résultat de l'évaluation est Vrai (ou différent de zéro pour le cas du C), l'instruction est exécutée

46

ETS
47

La syntaxe de l'alternative en C

- Plusieurs formats de l'alternative :

<pre>if (condition) { instructions; }</pre>	<pre>if (condition) instruction;</pre>
<pre>if (condition) { instructions; } else { instructions; }</pre>	<pre>if (condition) instruction1 else instruction2 ;</pre>
<pre>if (condition_1) instruction-1 else if (condition_2) instruction-2 ... else if (condition_n) instruction-n else instruction-else ;</pre>	

*Les accolades permettent de délimiter un bloc d'instructions dans la clause **then** ou la clause **else***

El Hachemi Aliakacem
INF147

47

ETS
48

Exemples

```
int x;
x = 5;

// Exemple 1 : attention
if (x > 2)
    printf("%d", x);
    x = 2 ;
```

Cette instruction sera toujours exécutée : pas d'accolades

```

// Exemple 2 : attention
if (x >= 2) {
    printf("%d", x);
    x = 2 ;
}

// Exemple 3 : attention
if (x = 8)
    printf("%d", x);
```

Cette instruction sera toujours exécutée : x=8 n'est pas une comparaison et retourne la valeur 8, la condition est évaluée comme vrai

El Hachemi Aliakacem
INF147

48

ÉTS

49

Exercice

- Écrivez un programme qui saisit un nombre entier et qui affiche un message si le nombre est négatif

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int nombre;
    printf("Entrez un nombre svp : ");
    scanf("%d", &nombre);

    if (nombre < 0)
        printf("Votre nombre est négatif\n");

    return EXIT_SUCCESS;
}
```

El Hachemi Aliakem

INF147

49

ÉTS

50

Sélection avec la clause sinon (*else*)

- La sélection en C

```
if (expression) {
    // Bloc d'instructions à exécuter si l'expression est évaluée à
    // Vrai (ou différente de 0)
}
else {
    // Bloc d'instructions à exécuter si l'expression est évaluée à
    // Faux (ou égale à zéro)
}
```

- La règle pour les accolades s'applique encore à la clause else : si les accolades sont manquantes, seule la première instruction fait partie de la clause sinon

El Hachemi Aliakem

INF147

50

ETS

51

Exercice

- Écrivez un programme qui saisit un nombre entier et qui affiche un message indiquant si le nombre est pair ou impair

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int nombre;
    printf("Entrez un nombre svp : ");
    scanf("%d", &nombre);

    if (nombre % 2 == 0)
        printf("Votre nombre est pair\n");
    else
        printf("Votre nombre est impair\n");

    return EXIT_SUCCESS;
}
```

El Hachemi Alikacem

INF147

51

ETS

52

Sélections imbriquées

- Nous avons la possibilité de mettre plusieurs instructions de sélection imbriquées (une à l'intérieur de l'autre)
- Exercice** : Écrire un programme qui saisit un nombre entier, puis indique en affichant un message, si le nombre est pair ou impair et, dans chaque cas, s'il est négatif ou positif

El Hachemi Alikacem

INF147

52

ETS

53

Sélections imbriquées

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int nombre;

    printf("Entrez un nombre svp : ");
    scanf("%d", &nombre);

    if (nombre % 2 == 0) {
        printf("Votre nombre est pair ");
        if (nombre < 0)
            printf("négatif");
        else
            printf("positif");
    } else {
        printf("Votre nombre est impair ");
        if (nombre < 0)
            printf("négatif");
        else
            printf("positif");
    }

    return EXIT_SUCCESS;
}
```

El Hachemi Aliakem

INF147

53

ETS

54

Sélection

Indication importante :

- Parfois, nous avons des instructions identiques qui se répètent dans les clauses *if* et *else* – Ces instructions sont indépendantes de la condition, on peut les sortir et les mettre avant ou après (selon le cas) la conditionnelle. Cela permet d'éviter la répétition inutile de code

➤ **Attention** : cette transformation ne doit pas changer la logique du code

El Hachemi Aliakem

INF147

54

ÉTS

55

Sélection

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int nombre;

    printf("Entrez un nombre svp : ");
    scanf("%d", &nombre);

    if (nombre % 2 == 0)
        printf("Votre nombre est pair ");
    else
        printf("Votre nombre est impair ");

    if (nombre < 0)
        printf("négatif");
    else
        printf("positif");

    return EXIT_SUCCESS;
}
```

El Hachemi Aliakem

INF147

55

ÉTS

56

Sélections multiples : Si – Sinon - Si

- La forme générale est la suivante :

```
Si condition-1
    // instructions à exécuter si condition-1 est vraie
Sinon Si condition-2
    // instructions à exécuter si condition-2 est vraie
Sinon Si condition-3
    // instructions à exécuter si condition-3 est vraie
...

Sinon
    // instructions à exécuter si aucune des conditions n'est vraie
Fin
```

El Hachemi Aliakem

INF147

56

ETS

57

Sélections multiples : Si – Sinon - Si

- Exemple :

```

Si j'ai au moins $30.00
    Je vais au restaurant
Sinon Si j'ai au plus $15.00
    je vais au cinéma
Sinon Si j'ai au plus $5.00
    Je vais à la crèmerie
Sinon
    Je reste à la maison
Fin
  
```

El Hachemi Alikacem

INF147

57

ETS

58

Sélections multiples : Si – Sinon - Si

Exercice :

Écrire un programme qui saisit deux nombres entiers représentant le numérateur et le dénominateur d'une fraction. Si le dénominateur est égal à 0 on affiche « impossible ». Sinon si le dénominateur est égal à 1 on affiche seulement le numérateur. Sinon si le numérateur est égal à 0 on affiche 0. Sinon si le dénominateur est négatif, on multiplie les deux entiers par -1 avant d'afficher numérateur/dénominateur

(solution : fraction.c)

El Hachemi Alikacem

INF147

58

Instruction itérative

59

ÉTS

60

Instruction itérative

- Les boucles permettent de répéter une série d'instructions selon l'évaluation d'une expression « habituellement » booléenne (une condition)
- Forme générale :
`Tant que` expression booléenne
tâches à répéter tant que l'expression est vraie
`Fin`
- Exemple :
`Tant que` Il ne m'a pas payé
je ne lui parle plus
`Fin`

El Hachemi Aliakem

INF147

60

ÉTS
61

La boucle while en C

- La forme générale de la boucle while

Boucle avec une instruction	Boucle avec un bloc d'instructions
<pre>while (expression) instruction</pre>	<pre>while (expression) { bloc d'instructions }</pre>

- Exemple :

```
int x = 5;
while (x > 0) {
    printf("%d " , x);
    x--;
}
```

Réponse : 5 4 3 2 1

El Hachemi Aliakem
INF147

61

ÉTS
62

La boucle while en C

- Attention à la boucle infinie – Le cas où la condition est toujours **Vrai**

Exemple :

```
int x = 5;
while (x < 10) {
    printf("%d" , x);
    x--;
}
```

x sera toujours inférieur à 10

El Hachemi Aliakem
INF147

62

ÉTS
63

La boucle while en C

- Exercice : Dites ce que vaudront les valeurs de x, y et z après l'exécution de cette boucle

```

int x = 5;
int y = 3;
int z = 3;
while (y) {
    x %= z++ ;
    y = y -1 ;
}
  
```

x = 2
 y = 0
 z = 6

El Hachemi Alikacem
INF147

63

ÉTS
64

Exercice

Écrivez un programme qui saisit un nombre entier et qui affiche tous les nombres de 1 jusqu'à la valeur entrée.

```

#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int nombre;           // nombre à saisir
    int tour;             // sert à compter les tours de boucle

    printf("Entrez un nombre svp : ");
    scanf("%d", &nombre);

    //affiche toutes les valeurs prises par la variable tour
    tour = 1;
    while (tour <= nombre) {
        printf("%d ", tour);
        tour++;
    }
    return EXIT_SUCCESS;
}
  
```

←

←

←

initialisation

Condition d'arrêt de la boucle

Convergence vers la condition d'arrêt

El Hachemi Alikacem
INF147

64

ÉTS
65

Bloc d'instructions

- Il n'y a pas de restrictions sur ce que peut contenir un bloc d'instructions : d'une clause *if*, *else* ou d'une boucle, etc., on peut y mettre n'importe quel type d'instruction :
 - Des affectations
 - Des appels de fonction d'affichage ou de lecture du clavier
 - des instructions de sélections
 - Des boucles
 - Etc.
- Permettant ainsi d'imbriquer des instructions de sélection et d'itérations dans d'autres instructions de sélections et/ou d'itérations

El Hachemi Alikacem
INF147

65

ÉTS
66

Bloc d'instructions

Exemple : écrire un programme pour la validation d'un nombre entré qui doit être entre 1 et 10

```

#define MIN 1
#define MAX 10
int main(void) {
    int nombre;

    //première sollicitation du nombre
    printf("\n\nEntrez un nombre entre %d et %d svp : ", MIN, MAX);
    scanf("%d", &nombre);

    //tant que le nombre est invalide, on avise et on redemande
    while (nombre < MIN || nombre > MAX) {
        if (nombre < MIN)
            printf("\n\nDésolé, mais le nombre doit être positif \n");
        else
            printf("\n\nDésolé, mais le nombre doit être inférieur ou égal à %d \n", MAX);

        printf("\n\nEntrez un nombre entre %d et %d svp : ", MIN, MAX);
        scanf("%d", &nombre);
    }
    printf("\n\nVotre nombre est valide \n");
    return EXIT_SUCCESS;
}
  
```

Condition d'entrée multiple

Structure if imbriquée

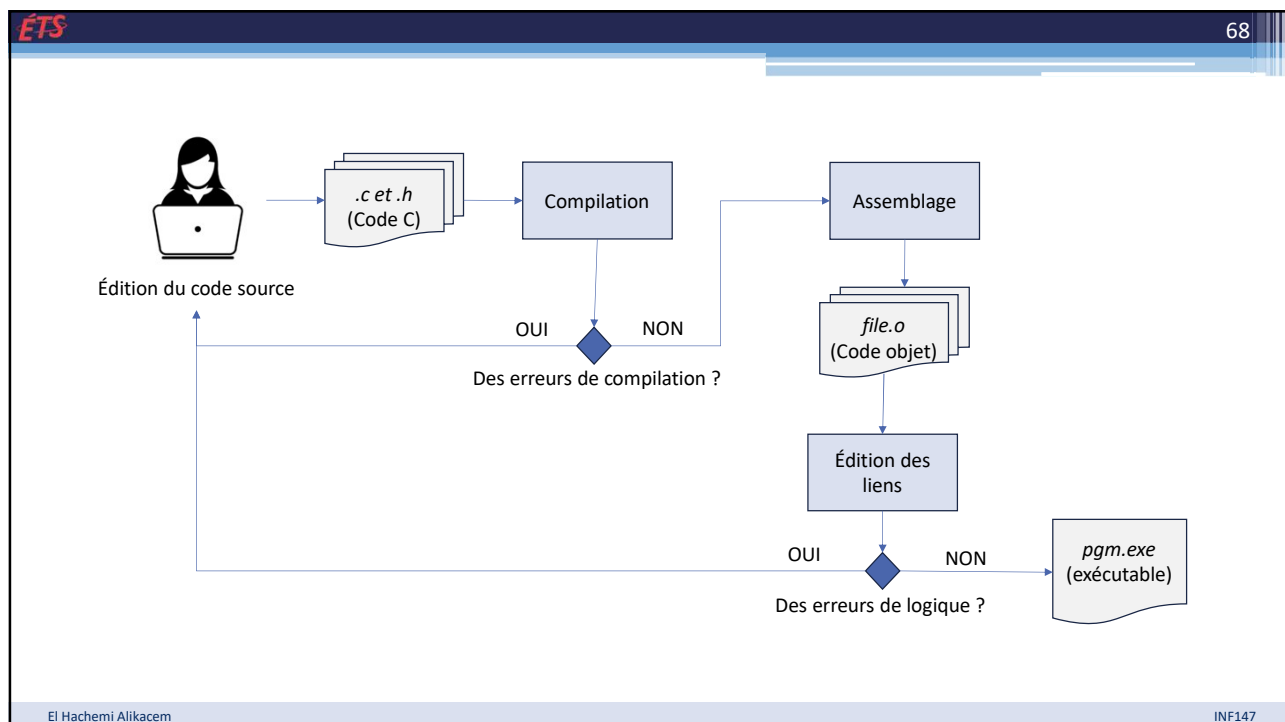
Contribue à la convergence vers la condition d'arrêt

El Hachemi Alikacem
INF147

66

Cycle de l'édition du code à l'exécution

67



68